Unsupervised Anomaly Detection System using Next-generation Router Architecture

Richard Rouil Nicolas Chevrollier Nada Golmie
National Institute of Standards and Technology
Gaithersburg, Maryland 20899

Abstract— **Unlike many intrusion detection systems that rely mostly on labeled training data, we propose a novel technique for anomaly detection based on unsupervised learning. We apply this technique to counter denial-of-service attacks. Initial simulation results suggest that significant improvements can be obtained. We discuss an implementation of our anomaly detection system in the ForCES router architecture and evaluate it using recorded attack traffic.**

I. Introduction

The research community has expressed much interest recently to better understand, model and mitigate Denial-of-Service (DoS) attacks [1] [2] [3]. A DoS attack may be characterized as an explicit attempt to exhaust key resources (*e.g.* network bandwidth, computing power, operating data systems structures) of the system under attack. Current defense mechanisms against DoS attacks consist of three components:

- **Prevention**: ingress filtering, anti-spoofing mechanisms.
- **Detection**: identifying anomalies and tracking suspicious traffic patterns.
- **Response**: rate limiting, filtering [4] or traceback [5].

Nevertheless, even if they are conceptually simple and are beginning to be well-documented, DoS attacks can usually bypass existing defense mechanisms through slight variations in their forms.

Edge networks often include Intrusion Detection Systems (IDS) in order to mitigate some forms of DoS attacks. Most of these IDS employ signature detectors [8], that scan incoming traffic, match it to predefined attack patterns, and raise adequate alarms in case of a hit. Subsequent actions taken by other components of the IDS include attack packet filtering, flow blocking, and traceback. In order for this technique to work, accurate characterizations of attack patterns must keep up with a multitude of DoS attack flavors and versions, which is problematic in practice.

Limitations of existing techniques have spurred interest in alternatives based on statistical learning. The basic thrust is to apply known learning techniques (both supervised and unsupervised) to network traffic in order to extract salient features and characteristics for attack traffic detection. Concurrently, advances in router architecture including the availability of next generation specialized network processors have made the implementation of these traditionally complex algorithms possible directly in forwarding hardware. Thus implementing powerful but computational intensive statistical algorithms on fast network processors holds the promise that we can perform packet inspection and forwarding at line speed and so to guard against the problem of DoS attacks.

In this paper, we propose a novel Anomaly Detection System (ADS). Like many such schemes, it functions by defining a baseline of normal or expected behavior and then determining if the current behavior observed deviates sufficiently from what is expected. However, unlike standard supervised learning techniques that imply the existence of a set of labeled training data, or a sequence of input/output pairs where the output is the desired classification, we use anomaly detection that does not require any *a priori* knowledge about the network [7] [10]. This is commonly known as unsupervised learning. Our concern is that training data, if available, is limited to past traffic trends that may not reflect the nature of future attacks. Our approach is to combine unsupervised learning techniques with anomaly system detection in order to create a robust mechanism to counter novel attacks. Our ADS works by grouping packets based on IP header fields into a small number of aggregates or so-called clusters. We show how tracking the behavior of these clusters over time can give additional clues about the likelihood of an attack. We investigate the implementation of ADS in the context of a network processor based router architecture and give the details of our design implemented on an Intel IXP 2800 development platform[1].

The outline for the remainder of this paper is as follows. Section II describes the proposed ADS and provides simulation results for its performance. Section III considers the implementation of ADS on a next-generation router architecture platform and discusses the performance results obtained from the hardware. The final section discusses future research directions.

II. Anomaly Detection System

In this section, we describe ADS, the proposed anomaly detection system, which consists of two main components, (1) a clustering algorithm, (2) and an anomaly detection scheme. We will start by describing the clustering algorithm and then highlight some techniques that can be used for anomaly detection.

[1] Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

A. *Clustering Algorithm*

Our goal is to aggregate the network traffic into a relatively small number of clusters, between 5 and 20. The clustering is done using the k-means algorithm [9], with batch processing for the training set. A n-dimensional vector (including the IP, TCP or UDP packet headers) represents each packet going on a network link. We use a splitting procedure to initialize the centroids in the algorithm. First, we calculate the centroid of the training packet sample $\mathbf{c}$, and then perturb it by a small vector ϵ to get centroids $\mathbf{c}+\epsilon$ and $\mathbf{c}-\epsilon$. Applying the k-means algorithm gives two optimized centroid $\mathbf{c}_1$ and $\mathbf{c}_2$. We then repeat the splitting and optimization process for a predefined number of iterations. We define the diffusion of a cluster by the mean distance between each sample of a given cluster and the centroid of this cluster. At each splitting step, this diffusion is compared to a given threshold. The less diffused a cluster is, the more compacted it is. If the cluster is compacted enough, no further splitting occurs. The algorithm uses the Manhattan distance measure defined as follows:

$$\|\mathbf{x}-\mathbf{y}\| = (\sum_{j=1}^{n} |x_j - y_j|), \tag{1}$$

In order to avoid scaling issues among different dimensions, each value is first normalized by its maximum value during the epoch. We have tested other distance measures, but the Manhattan distance, besides being as efficient as the others, appears to be the lightest in terms of computation time. The base algorithm initially introduced in [6] is described below.

```
Compute centroid of training sample;
Split centroid into 2 centroids;
Run k-means algorithm to optimize 2 centroids;
for N iterations do
    foreach centroid do
        if cluster associated is not sufficiently compacted
        then
            Split into 2 centroids;
            Run k-means algorithm to optimize 2
            centroids;
        end
    end
end
```

A merging procedure is performed at the end of the algorithm, in order to combine identical clusters, or clusters with close centroids. The diffusion, merge threshold, and the perturbation ϵ are chosen based on raw data traffic analysis and simulation results.

These clusters can be used as part of a passive strategy to attenuate DoS attacks even without explicit detection. Since similar packets will tend to cluster together, bad packets will likely end up in the same cluster. This assumption is particularly likely to be true for brute-force distributed DoS attacks where packets share similar characteristics such as a destination IP address or a packet size. Fig. 1 depicts the dynamics of a possible passive reaction. On each router output line card, clusters are mapped to different queues. Packets entering the system are compared to the centroids computed, so each packet mapped to centroid i ends up in queue i. By forcing the illegitimate or bad traffic to be clustered into one or a small number of clusters and then mapped into a small number of queues, a rate-limiting effect is accomplished.

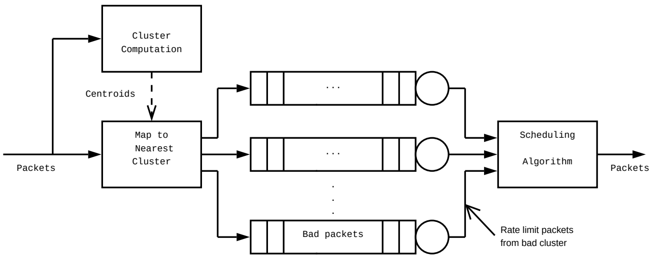

Fig. 1. DoS Attack Passive Reaction.

B. *Anomaly Detection*

As mentioned previously, clustering represents the first step in anomaly detection. The next step consists of tracking the changes of each cluster's centroid over time. Here, we consider the use of three different centroid characteristics to determine and track changes: compactness, number of samples per cluster, and cluster dimensions. We discuss each one in turn. The observations are based on analysis of several hundred traces of attack traffic collected from simulations and measurements of networks of varying size (50 - 500 nodes).

Compactness
Most brute force DoS attack flows analyzed are characterized by a low diffusion (dense flows). Thus, a low diffusion represents a partial attack indicator.

Sample size
Another indicator of attack traffic is a cluster with a very large size relative to other clusters detected during the same time period. Since most attack traffic has similar characteristics (*e.g.* the same destination address, port ID) it generally ends up in the same cluster.

Cluster dimensions
Finally, we observe that at the beginning and the end of an attack cluster dimensions such as the IP address, protocol number, packet size vary drastically as the mixture of packets in the network changes significantly.

All the observations discussed above should raise together and individually some type of an attack flag that requires further investigations on a specific cluster, activate filtering for traffic related to the suspicious cluster identified and assign strict queuing policies to strengthen the passive reaction.

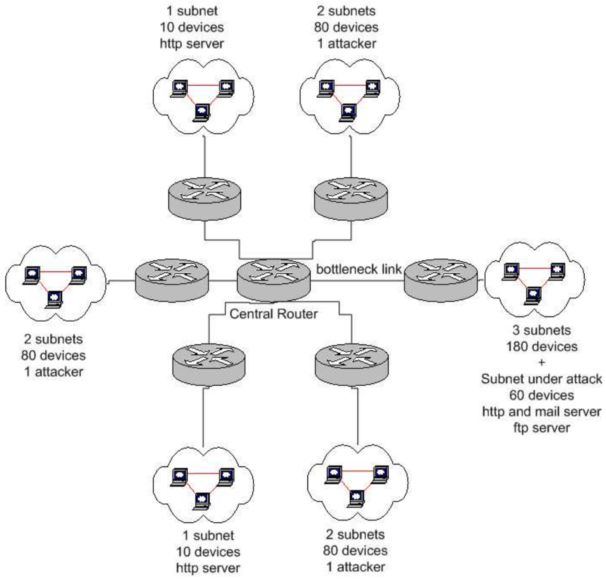

Fig. 2. Simple Test Network Topology.

C. *Performance Evaluation*

We evaluate the clustering algorithm presented earlier and show how the observations discussed previously can be used to effectively detect anomalies and potential DoS attacks. We use OPNET to simulate a UDP flood attack in a 500-node network.

1) Simulation Set-up: Fig. 2 shows the topology of the network simulated by OPNET. It consists of a total of 500 end-user devices, 11 subnets, 3 *http* servers, one *ftp* server and one mail server. The legitimate traffic is a combination of *http*, *smtp* and *ftp* packets arranged as follows. The major contribution comes from *http* which represents 90% of the total traffic in bits, followed by *smtp* and *ftp* traffic which contribute 5% each. Each application is characterized by a default OPNET profile and attributes. Similar to the attack model developed by Gregg et al [11], 3 attackers from 3 different subnets try to overflow the bottleneck link of the network by sending UDP packets from spoofed IP addresses. Out of the 10-minute simulation, the network is under attack for 4 minutes. We ran the simulation for a maximum of 16 centroids and each packet header is represented by a six-dimensional vector that contains: (1) a source address, (2) a destination address, (3) a protocol type, (4) a packet size, (5) a source port, and (6) a destination port. While mapping centroids to output buffers provides a passive reaction to most DoS attack types, further analysis of the clustering results can provide a more active response to a DoS attack.

2) Results Analysis: The analysis of the results reveals the following insights. During periods of no congestion, the bottleneck link is about 65% full on average, while during the congested period, the offered load reaches over 100 percent of capacity. In the case of n output queues, the length of each queue is 1/n of the length of the aggregate queue, so that there is no net increase in memory usage.

TABLE I
PERCENTAGE OF PACKETS DROPPED PER SCENARIO

	Legitimate traffic	illegitimate traffic
Single Aggregate queue	4.11	9.7
8 queues + passive reaction	1.11	18.1
16 queues + passive reaction	0.22	20.4

First, we analyze the results of the passive reaction. Table I shows the percentage of legitimate packets dropped (first column) and the number of illegitimate packets dropped (second column) when there is a single output queue (first row), 8 output queues combined with the passive reaction (second row) and 16 output queues combined with the passive reaction (third row). We note that the percentage of legitimate packets dropped decreases when the passive reaction is active. This percentage is relatively low because the average is done over the whole simulation and the loss occurs only during the period when the network is under attack. On the other hand, we observe that more illegitimate packets are dropped as the number of queues increases. As the illegitimate traffic ends up in a smaller proportion of the clusters, these clusters will contain a larger number of packets and will likely overflow more often, resulting in a higher packet drop rate.

Further analysis of the simulation results obtained allows us to verify some of the anomaly detection observations noted previously, concerning the sample size and the cluster compactness.

The normalized diffusion is shown over time in Fig. 3. For the sake of presentation clarity, we show two clusters (Cluster 1 and Cluster 2) out of the eight obtained in the experiment. Cluster 1 is related to legitimate traffic while Cluster 2 represents the attack traffic. Each of these clusters represent at least 10% of the overall bandwidth making their characterization statistically significant (the remaining clusters are either similar to Cluster 1 or have very little traffic associated with them and therefore are not significant to the results shown here.)

In Fig. 3 the diffusion of Cluster 1 varies around 2 and 4 until the attack is scheduled to take place around 250 seconds. At that time, we note a sharp dive of the Cluster 2 diffusion that goes down to 0. We also notice a slight increase in the overall fluctuations of the diffusion of Cluster 1 that is being affected by the sudden drop in Cluster 2 diffusion. We verify that a similar pattern is effecting other clusters that are similar to Cluster 1.

The very low diffusion value (which indicates that packets related to the cluster are very close to its centroid, thus that the cluster is extremely compact) exhibited by Cluster 2 indicates that the traffic being assigned to Cluster 2 has very similar or almost identical characteristics. While, a low diffusion by itself may not necessarily indicate attack traffic, this combined with an assessment of a large cluster should increase the likelihood of detecting an attack. Note that the slight increase in the diffusion for the other clusters is due to the clusterization algorithm adapting to the arrival of a a new traffic pattern,

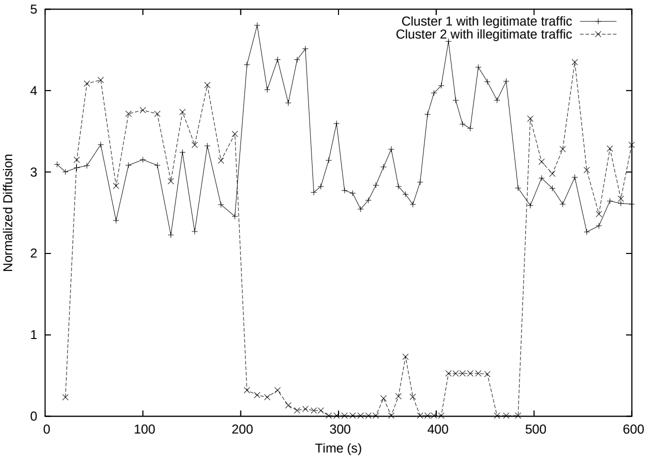

Fig. 3. Evolution of diffusion.

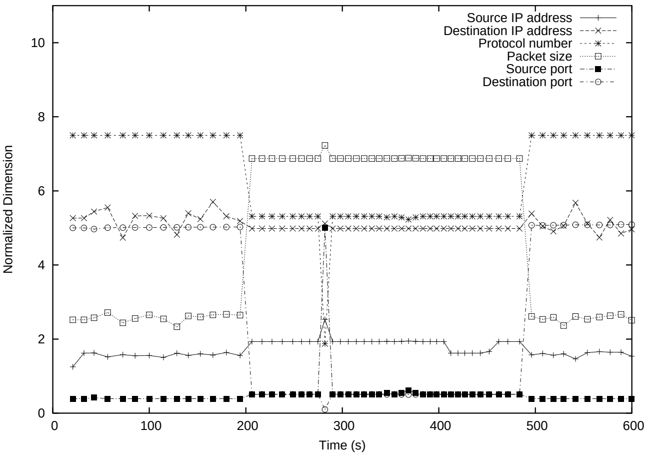

Fig. 4. Evolution of dimensions of Cluster 2.

which in this case represents the attack traffic.

We turn now to cluster dimensions. In Fig. 4, we plot over time the centroids of all six parameters that were used to characterize Cluster 2, the cluster containing attack traffic. In this case, we observe a significant variation in the cluster centroids coinciding with the attack that is occuring between 200 and 500 seconds. When the attack ends at 500 seconds, we note that the centroid values go back to their pre-attack values. This represents another strong indicator for the presence of attack or suspicious traffic.

In summary, we can identify at least three indicators for detecting the presence of suspicious or attack traffic. Since we are mostly interested with attacks that involve large flows and occupy large percentage of the bandwidth available, we therefore use the flow or sample size as the first indicator. We can then look at cluster compactness that indicates how packets in the same cluster are related to each other. A large number of packets that are very similar can point to a brute-force-type attack where packets have very similar characteristics. Finally, we find that tracking the cluster dimensions, which are the parameters used in the clusterization, and their variations over time can be another effective indicator. While the sample size forms the basis for any detection strategy, we see a combination of compactness and dimension variations as fairly effective in detection attack traffic.

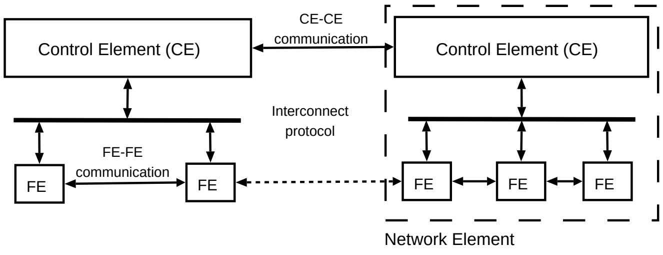

Fig. 5. Next-Generation Router Architecture.

III. Hardware Implementation

The emergence of new router architectures separating control and forwarding planes as shown in Fig. 5 that separates the control and forwarding planes facilitates the distribution of control information such as intrusion detection notifications and alarms across a wide area network. This represents a step towards the deployment of effective distributed protocols for intrusion detection. In this new architecture, the Network Element (NE) is logically separated into one or more Control Elements (CEs) and one or more Forwarding Elements (FEs).

In this paper we focus on the FE and the availability of highly specialized network processor units such as the Intel IXP 2800 in order to implement the computationally intensive clustering algorithm that was described previously.

A. Clustering Algorithm Implementation

In order to implement the clustering algorithm in hardware, we use the Intel IXDP2800 Advanced Development Platform. This is composed of two IXP2800 network processors, and is designed to support applications with traffic up to 10 Gbit/s. Each network processor contains a fast path that uses highly optimized microengines, and an XScale core for additional packet processing. Ideally, to achieve high speed packet forwarding, packets processing should remain in the microengines. Fig. 6 shows the main elements of the NP platform that are relevant to our system.

1) Traffic going through the network processor is sampled and specific parameters such as the IP address, port number, source and destination addresses, and protocol IP, are extracted from each packet. Due to the very high speed traffic and because the parameter extraction occurs in the NP core, our implementation can achieve only a 10% sampling rate of the overall traffic seen. Although this may represent a limitation in terms of the statistics collected, 10% of a 1 Gbit/s link still represents a lot of traffic and should provide a good approximation for the overall traffic.
2) The XScale core operates in two phases. In the first phase, samples are extracted by the microengines. Once a predefined training set number is reached, the core

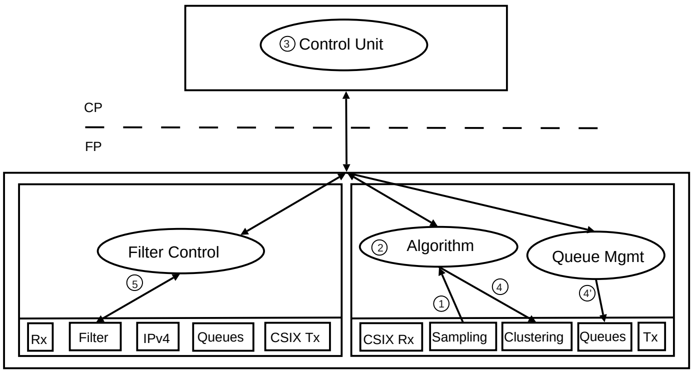

Fig. 6. Anomaly detection system overview.

stops collecting data and starts computing the cluster centroids.

3) At that point, the Control Unit receives information about the clusters from the core and decides on the next actions to follow. This unit can be either local or remotely connected. We have used a simple co-located control unit for our experimentation.

4) & 4') The Control Unit decides if the centroids need to be updated. It may also interact with the queue management to apply policies on the queues. The queue scheduler implements a Deficit Round Robin (DRR) algorithm on the queues of each port. The Control Unit limits the number of credits available to each queue thus rate-limiting bad traffic.

5) A filtering mechanism can be installed in the ingress side of the platform in order to stop suspicious traffic before it enters the router. Our work has been concerned with the clustering algorithm and has not yet included any input filters.

B. Testbed and Hardware Performance Results

Our experiment environment consists of the IXDP2800 platform running with VxWorks on the XScale core. This platform provides an interface with 10x1-Gigabit Ethernet ports. To generate traffic, we use 4x1-Gigabit Ethernet ports from a Smartbits 6000B Performance Analysis System. As shown in Fig. 7, we use three ports to generate traffic and one to monitor and received the routed packets. The test traffic consists of the following:

- *Normal TCP*: 150 TCP flows totaling 800 Mbit/s.
- *Normal UDP*: 50 UDP flows totaling 150 Mbit/s.
- *Attack UDP*: 5 UDP flows representing 400 Mbit/s.

The proportion chosen for the mix of *Normal TCP* and *Normal UDP* corresponds to realistic Internet traffic where a ratio of TCP to UDP traffic is typically 8 or 9 to 1 (Note that this ratio can vary slightly between different network types without affecting our results). While for both *Normal TCP* and *Normal UDP* flows the parameters used for the source and destination addresses, the source and destination ports, and packet sizes were different, all 5 *Attack UDP* flows had the same destination address, port and packet size. Only the source parameters for the *Attack UDP* flows are varied to emulate a distributed 5-node UPD flood attack. Also note that the *Attack UPD* traffic consists of a relatively low number of flows that use a lot of bandwidth.

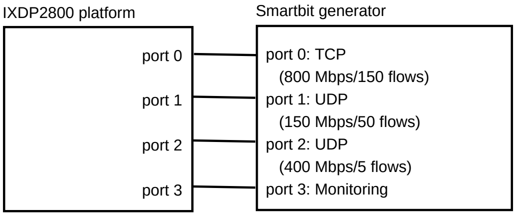

Fig. 7. Testbed setup.

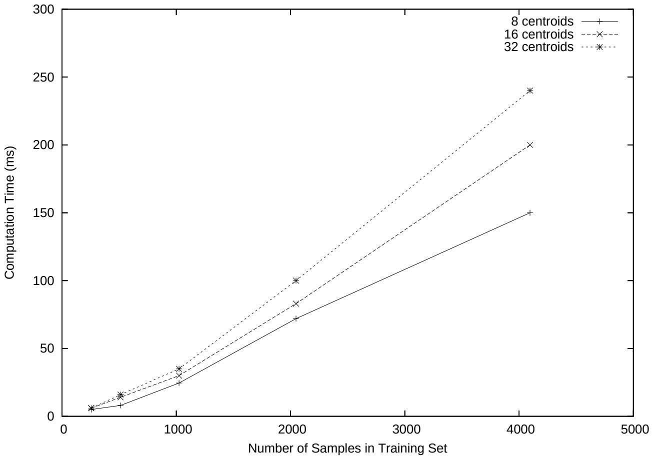

Fig. 8. Algorithm Performance.

Next we will discuss the hardware performance of the clustering algorithm in terms of computation speed and packets dropped in the output queues.

1) Computation time: The real implementation allows us to quantify the time it takes to compute the clusters. This time depends on the number of samples collected and the number of centroids the algorithm computes. Fig. 8 shows the computation time for a training set of 256, 512, 1024, 2048, and 4096 packets, for 8, 16, and 32 centroids.

With training set sizes less than 2048, the computation time remains less than 50 ms regardless of the number of centroids used. For larger training set sizes, the computation time increases with the number of centroids used. The difference in computation time for 8 and 32 centroids is around 90 ms for a training set of 4096. We use a training set of 2048 and 16 centroids, which represents a good trade-off between the implementation computation time and the accuracy of the clustering results.

2) Queue Management Policy: As described previously (II-A), by mapping each cluster to an outgoing queue and setting a queue management policy to rate-limit outgoing traffic based on information from the anomaly detection control unit, we can effectively mitigate a DDoS attack that generates a large amount of traffic. Since attack traffic ends up in the

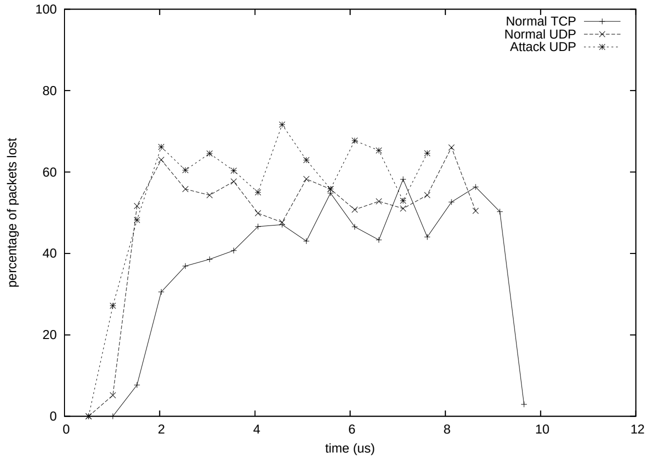

Fig. 9. Packets lost per class for a single queue.

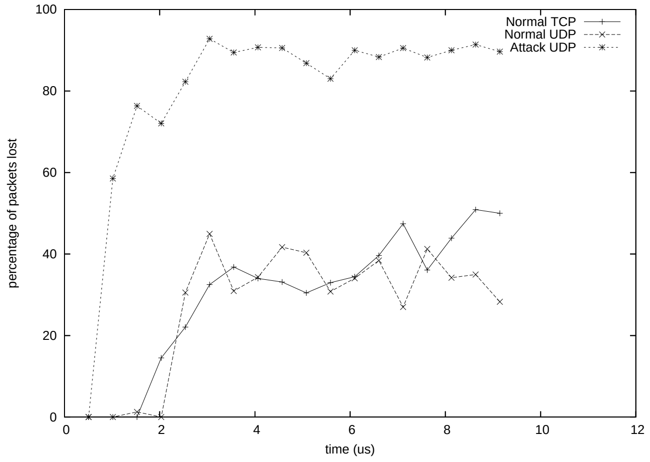

Fig. 10. Packets lost per class using clustering algorithm and 16 queues.

same cluster and is mapped to a single queue, attack traffic is dropped more heavily.

Fig. 9 and Fig. 10 highlight the performance of this sort of passive strategy in the anomaly detection system by showing the number of packets dropped per traffic flow type. Fig. 9 shows system behavior without ADS using one output queue size equal to 96 packets and a bandwidth of 1024 bytes per round. Fig. 10 shows the behavior of the ADS using the clustering algorithm with 16 clusters mapped to 16 queues of size 6 packets, and a bandwidth of 64 bytes per round. In Fig. 9 the packet loss is around 60% for all traffic types, while in Fig. 10 the *Attack UDP* flow incurs a much heavier packet loss rate of 90%. The other traffic types are less affected with packet loss of around 35%.

IV. Conclusion and Future Work

In this article, we have demonstrated the use of packet clustering based on unsupervised learning to effectively mitigate DDoS attacks. Based on characteristics extracted from sampled traffic, we can derive criteria that can be used to trigger anomaly detection alarms. These criteria include packet similarities and how they are related to one another, the amount of traffic sharing the same relations, and the evolution of these characteristics over time.

We propose a suitable packet clustering algorithm and implement it in simulation and on a network processor platform. This shows the implementation feasibility of such computationally intensive algorithm by taking advantage of the ability of next-generation routers to perform computation in the forwarding plane and at line speed while other more intelligent tasks to analyze and derive information from traffic monitoring occurs in the control plane. In our implementation, packet header information is extracted and compared to current clusters characteristics (centroids) at line speed, while those centroids are only updated periodically in the background.

We show that combining this clustering mechanism with a simple queue management policy that limits the size of the output buffers, leads to a selective packet dropping that mainly targets attack traffic. Additionally, we believe that anomaly detection triggers can be used to adaptively devise these queue management policies.

Future work will include implementation of filter controls and distributed mechanisms to share clusters information between several nodes to enable concerted actions.

References

[1] J. Mirkovic, J. Martin and P. Reiher, "A Taxonomy of DDoS Attacks and DDoS Defense Mechanims," *UCLA CSD Technical Report CSD-TR-020018.*

[2] A. Hussain, J. Heidemann and C. Papadopoulos, "A Framework for Classifying Denial of Service Attacks," *ACM SIGCOMM'03*, Karlsruhe, Germany, August 2003.

[3] J. Mirkovic, "D-WARD: Source-End Defense Against Distributed Denial-of-Service Attacks," *PhD thesis*, 2003.

[4] R. Mahajan, S. M. Bellovin, S. Floyd, J. Ioannidis, V. Paxson, and S. Shenker, "Controlling high bandwidth aggregates in the network," *Computer Communications Review*, Vol. 32, Num. 3, July 2002.

[5] S. Savage, D. Wetherall, A. Karlin and T. Anderson, "Pratical Network Support for IP traceback," In *proceedings of the ACM SIGCOMM Conference*, pages 295-306, Stockholm, Sweeden, August 2000.

[6] N. Chevrollier and R. E. Van Dyck, "Packet Filtering for aggregate-based congestion control," *Proc. CISS*, Princeton, NJ, March 2004.

[7] S. Zanero and S. M. Savaresi, "Unsupervised learning techniques for an intrusion detection system", *Proceedings of the 2004 ACM symposium on Applied computing*, Nicosia, Cyprus, 2004.

[8] V. Paxson, "Bro: A System for Detecting Network Intruders in Real-Time", *Computer Networks*, 31(23-24), pp. 2435-2463, December 1999.

[9] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning Data Mining, Inference, and Prediction*, Springer Verlag, 2001.

[10] E. Eskin, A. Arnold, M. Prerau, L. Portnoy and S. Stolfo, "A Geometric Framework for Unsupervised Anomaly Detection: Detecting Intrusions in Unlabeled Data," *Data Mining for Security Applications*, Kluwer, 2002.

[11] W. J. Blackert, D. M. Gregg, A. K. Castner, E. M. Kyle, A. L. Hon, and A. M. Jokerst, "Analyzing interaction between distributed denial of service attacks and mitigation technologies," *Proc. DISCEX III*, Washington, D.C., April 22-24, 2003.